我经历的重构

潘俊勇

Pycon China 2013

关于我

- 潘俊勇，老潘
- @panjunyong
- Zope/Plone社区：2002-2006
- 易度云办公(everydo.com)：2007-至今

Limodou：重构，Refactoring

回 潘俊勇:所以看来大多数的讨论都是似是而非。//@易度-潘俊勇:我自己定义的是 现有功能的重写，就算重构。重构可能增加新功能，也可能不增加。这是我们大多数人的理解。可能和这个不一致。

@limodou

分享好文：【什么是重构，什么不是重构 | 外刊IT评论网】:重构(Refactoring)这个词最初由Martin Fowler 和 Kent Beck给下的定义，它是 一种修改，使软件的内部结构更容易理解，在不改变软件的可见行为方式前提下使软件更容易变更...它是一种有节制的 （分享自 @外刊IT评论网） http://t.cn/zOEc3BR

12月5日 20:13 来自外刊IT评论网 | 转发(5) | 评论(3)

12月5日 22:06 来自FaWave | 举报 | 转发(1) | 收藏 | 评论(1

为什么需要重构

- 有味道的脏代码
  - 毒害新人：学到坏东西
  - 影响老手：心情不佳
- 重构
  - 建立清新美好的代码环境
  - 激发追求美好的欲望

什么时候重构

- 看到问题就重构
- 员工离职的时候，和交接人员重构
- 小的重构，多多益善

减少重构

- 设计 review
- Code review
- 新员工培训
- 手把手改代码
- 及早优化开发框架
- 更多的文档

这是我想讲的吗？

No

我更想说的是“大重构”

- 人在江湖漂，哪能不挨刀

- 我和“那些坑”的故事
  - 大的模块全部重写
  - 更换架构、框架，甚至语言
  - 技术选型碰到的问题

为什么“大重构”

- 旧代码：已经让人犯恶心
- 无法适应新需求
  - 性能更快
  - 体验更好
  - 移动端等的支持
- 新技术成为新标准：
  - 前端：ajax框架、mvc、Bootstrap/html5
  - 后端：新语言、云平台、开放架构

渐进，或者革命?

案例一：ZOPE2 → ZOPE3

Zope2的繁华（1997-2004）

- Zope从python社区的明星
- Zope公司200多人
- 社区庞大活跃，健全的生态
- 活跃的Plone
- 《魔法大熔炉》经典成案例

Zope2的优势

- 浏览器上web开发
  - 非常快速
  - 非常简单
- 现在Salesforce提供的PaaS也是这样

Zope2的问题

- 浏览器开发很原始
  - 代码如何版本管理
  - 浏览器开发模式不清晰
- **底层代码耦合高，基于继承，比较复杂，难维护和扩展**

当年的明星: java（2000年）

- J2ee太酷了
- 组件架构很酷
- 基于接口编程很快
- Xml配置应用很酷！

Zope3的想法：干点大事了

- 重构！完全重构
- 向偶像学习，打造python版本的j2ee！
- 为python写组件架构ZCA：zope组件架构
- Xml装配组件
- 基于接口，全部重写!

后来。。。

- 首先偶像J2EE因为太复杂，并没有保持偶像地位
- zope3也很复杂
- 和zope2完全不兼容
- Zope2被迫引入了Zope3, 更复杂
- 推出时间太长，4年多
  - zope2停止发展，问题无人解决
  - 大量竞品出现:Django、ROR
- 摧毁生态，社区迷茫
- 社区基本瓦解，Zope基本死亡，Plone停止发展
- Zope公司缩小10倍，20多人

Zope2重构反思

- 不能抛弃用户
  - 首先优化浏览器开发模式（解决用户之痛）
- 底层代码渐进演化
  - 应该采用wsgi分割的方式来做
  - 将复杂系统物理分割为多个子系统
  - 另外一个项目：Repoze

迷失的黑客

- 给j2ee坑了
- 妄自菲薄：我们怎么写了如此shi的程序
- 程序员着急了
  - 我是不是该干点大事了！

重构：渐进，或革命

- 渐进：
  - 如同改革，不断局部改良，确保大局稳定
  - 风险最小
  - 不会影响整个生态
  - 需要时间和耐心，找到通道不容易
- 革命：不得已的选择
  - 休克疗法
  - 失败风险大
  - 一定有牺牲

内部必须渐进
革命来自外部

Perl 6 的段子

- Perl 5源代码很复杂，新开发人员很棘手
- Python蚕食Perl市场
- 宏伟的、革命的Perl 6
- Perl社区的混乱

Perl6 诞生

2000 年 7 月 19 号，在一场无聊沉闷的政治组织会议中，Jon Orwant 站在一边听了几分钟，然后很淡定的走向咖啡桌，拿起一个个杯子往对面的墙上扔去，边扔边说"我们得想办法激励社群，不然都完了，大家越来越无聊，都去做别的事了。我不管您们怎么做，可您们得搞些大事出来。"，然后他掉头就走，这场事件，触发了 Perl6 诞生的火苗。经过十年的磨砺，2010 年 7 月 29 号，Perl6 的第一个实现版 Rakudo Star 终于发布了，这头曾经背负重担的骆驼轻装上路了。

Python: 2-> 3

- 类似的问题：
  - Python2长久不可能消亡
  - 实际需要兼容2种语法
- 也不算成功

好吧，看看我的革命史

案例二：易度文档管理 1.0 -> 2.0

易度 1.0

- 7年前
- 基于Plone 3开发
- 已经产品化
  - 安装程序
  - 文档
  - 网站
- 已经卖钱了

问题

- Plone的GPL许可限制商业化
- Plone 3是个庞然大物
  - 历史痕迹(zope2 + zope3)，内部很脏
  - 我自己操控都难
  - 别提新人了
  - 都不喜欢Plone

闪闪明星：zope3

- Launchpad也是zope3开发的
- 出现一些zope3开发的很酷的网站
- 社区都在想如何迁移到zope3

我也要干一票大事

- 放弃易度1.0
- 专心使用“美好的”zope3重构

巨大的代价

- Zope3都不懂
- 文档少
- Zope3并不是从前的那么美好：复杂
- 招人同样难上手
- 还是各种坑，代码乱得…
- 一年多才出产品
- 没有收入，销售给开发巨大压力！
- 焦虑的技术负责人：压力重重，各种加班
- Zope3接近死亡
- 易度成为全世界最大的zope3应用
- 正在漫长的去zope3化

错在哪里

- 不要盲目相信宣传，自己试验评价
- 不可冒进
- 面对自己不那么酷的技术，不能妄自菲薄

有更好的重构之路吗

- 回归到cmf
  - CMF是Plone的底层
  - CMF代码比较少，容易维护
  - CMF没有许可问题
- Zope2其实有另外一条演变之路
  - Repoze: 基于wsgi简化
  - 最终可能过度到bfg，pyramid

回顾Yammar： java-scala-java

希望更快的开发效率

Back in August 2010, Coda said on the Yammer Engineering blog that they were moving to Scala for their realtime future. The goal was to continue running on the JVM (for performance reasons) and that the conversion had resulted in approximately a 50% code reduction:

> Our initial prototype of Artie was in Java, but as a weekend experiment I tried reimplementing it in Scala 2.8. After a day, I had dropped about half the lines of code and added several tricky features. I was sold. It might be easier to hire Java developers, but a Scala team will be able to get a lot more done

Fast forward a year and a quarter later, and the decision is being reversed:

> Right now at Yammer we're moving our basic infrastructure stack over to Java, and keeping Scala support around in the form of façades and legacy libraries. It's not a hurried process and we're just starting out on it, but it's been a long time coming. The essence of it is that the friction and complexity that comes with using Scala instead of Java isn't offset by enough productivity benefit or reduction of maintenance burden for it to make sense as our default language. We'll still have Scala in production, probably in perpetuity, but going forward our main development target will be Java.

Scala太复杂了，不容易维护

http://www.infoq.com/news/2011/11/yammer-scala

插播新闻

是不是paypal换了个技术主管啊，搞这事

@InfoQ

【PayPal从Java切换到JavaScript】 PayPal已经决定使用JavaScript开发Web应用程序，从浏览器一直到后端服务器，并放弃了使用JSP/Java编写的遗留代码。http://t.cn/8k5WB9m by 马德奎

12月4日 12:16 来自InfoQ (38) | 转发(343) | 评论(51)

12月4日 12:31 来自新浪微博 阅读(757) 推广 | | 转发 | 收藏 | 评论

该怎么选型：通用技术，专门技术

案例3：曾经的KSS、BEAKER

Plone： KSS → jquery

- 当年是ajax的战国时期
- Jquery当时还不出挑
- 很多程序员不懂javascript
- 于是自制了KSS：
  - 不用懂javascript，就能ajax
  - Kss类似css，描述型的语法
  - 增强kss可以写插件
  - 和底层的ajax框架隔离

KSS的问题

- 性能问题
- 上手简单，但是深入比较难
- Kss不能解决所有问题
- 复杂的需要写插件

- （让我想起对angularjs的评价）

KSS选型警示

- 没有万能药
- 通用性问题：
  - 和性能对抗
  - 和可维护性对抗
- 警惕“描述性语言”的宣传
  - 这个可能只是哄哄初级用户
  - 和灵活性是对抗的
  - 现在css都变得动态了！

KSS→jquery：完全重写？

- 是的Plone花了几个版本来做这个事情
- 易度找到了演进的路子
  - 总结了几种交互模式
  - 使用jquery简化和改造了KSS
  - 服务端基本无需重写（继续python的方法ajax）
  - 前端只需要稍微调整
  - 解决性能问题！
- 一个版本、一个人，完成平滑迁移！

Session选型： Beaker

Lazy-Loading Sessions: No performance hit for having sessions active in a re
actually used

Performance: Utilizes a multiple-reader / single-writer locking system to pr
effect when caching.

Mulitple Back-ends: File-based, DBM files, memcached, memory, and datab
back-ends available for sessions and caching

Cookie-based Sessions: SHA-1 signatures with optional AES encryption for
based session storage

Flexible Caching: Data can be cached per function to different back-ends, wi
expirations, and different keys

Extensible Back-ends: Add more backends using setuptools entr points to
ends.

万能药？

深入beaker的问题

- 我们希望用redis来管理session
- 官方没有redis的backend！
- 自己写！
  - beaker代码很复杂，难于理解

后来

- 我们对session的管理要求更多
- 需要了解Beaker深入细节
- 代码。。。
- 抛弃beaker，自己写了基于redis的session管理，并不复杂

总结

- 通用性的万能药
  - 宣传大而全
  - 内部实现复杂化
  - 操控度会不灵活
- 简单的、单一的方案可能更好

从索取到贡献的改变

案例4：易度队列的折腾

最早的选择

- 需要更好的异步处理方法
- 选择zc.async
  - 强大的功能
  - 完善的文档
  - 基于twisted。。。
- 但是
  - 不稳定，twisted不了解，实现很复杂
  - ZODB大量冲突

好吧，换一个简单的

- z3c.remotetask
- 卡死的问题没有了
- 但是ZODB大量冲突

直到看到:

- 队列系统，不能基于数据库来实现
- 否则一定会大量冲突

队列选型

- Rabbitmq
- beanstalkd
- 基于redis的队列

Celery?

- http://www.celeryproject.org/
- Python编写
- 支持redis
- 但是有万能药的伟大理想：

MULTI BROKER SUPPORT

The recommended message broker is RabbitMQ, but support for Redis, Beanstalk, MongoDB, CouchDB, and databases (using SQLAlchemy or the Django ORM) is also available.

害怕大而全的通用型方案

- 一年被蛇咬，十年怕草绳
  - 不再相信开源宣传
- 希望更简单
- 希望更可控
  - 我们对队列有很多要求

最后：自制ztq队列

- 自制的、开源！
- pyramid
- 用得很爽
- 各种特性自由加
- 第一个自己比较满意的产品

https://github.com/everydo/ztq

ZTQ：Z Task Queue

ZTQ是python语言的一个开源异步队列服务, 使用redis作为队列的存储和通讯。

和其他队列服务不同，ZTQ的设计目标是：

- 实现简单
- 容易使用
- 可靠
- 错误、拥塞时，可管理
- 容易调试
- 灵活调度，高效利用服务器

详细介绍可参看： https://github.com/everydo/ztq/raw/master/about-ztq.pptx

ZTQ是由易度云办公(http://everydo.com) 赞助开发的，在易度云查看和易度文档管理等系统中广泛使用。

这个故事告诉我们

- 被坑多了，你也会有自己的审美
- 你可能做出更好的东西

- 面对开源
  - 我们不仅仅是索取
  - 动手去贡献！

再一次开源

案例5：易度索引的演变

易度索引的问题

- ZODB用起来很爽
- 但，ZODB自身没有索引
- 社区的zcatalog性能很慢！

借助关系数据库？

- 对多条件查询，性能也不高
- 全文索引性能不够
- 很难做到schemaless
  - 动态新增字段

选择xapian/xappy

- 虽然是一个搜索引擎
- 但是xappy等让xapian也支持字段
- 搜索很快
- Python支持很好！

问题

- Xappy不支持任意索引字段schemaless
  - 自定义表单，需要特定的搜索
- 大数据量支持不好
  - 不支持分区
- Xappy项目几乎停止了！

再次开源：zapian

- 抛弃xappy
- 借鉴elasticsearch的schemaless的接口
- zapian
  - 使用更简单
  - 无需定义字段，schemaless
  - 支持分区

用起来很爽

- 我们自己独特的组合：
  - ZODB + xapian
- 上层搜索我们提供类似django的接口
- 使用方便，开发高效

Xapian还有很多坑

- 某些情况内存可能泄漏
- 不是真正的MVCC
  - 写很快的时候，和读冲突
- 偶尔数据库崩溃，无法修复
- Xapian几乎是一个人的项目
- Replication方案很复杂

面对这个坑

- 我们准备
  - 先看看能不能解决xapian内存泄漏的问题
  - 未来可能切换到lucene
- 人在江湖漂
  - 谁没有上错几次贼船

我们自己的一次万能药尝试

案例6：兼顾ZODB和RDB存储？

易度运营平台的存储选型

- ZODB实现很简单
  - 但统计分析方面可能是问题
- 关系数据库擅长数据统计分析
  - 写起来可能啰嗦些
- 考虑兼容ZODB和rdb两种存储

选择谁？

- 先试试关系数据库，不行切换ZODB
- Java设计模式最擅长这套
  - xxxManager
  - xxxDAO
- 好吧引进这套

放弃关系数据库

- 业务逻辑非常复杂
- 引入20多个表，相互关联
- OR太非人类了
  - 对于习惯ZODB开发的人

及时切换回ZODB

- 速度大大加快!
- 但是那些Java风格DAO
  - 很多多余代码，让人窒息
  - 比较难看懂

不要大而全

- 存储层隔离，属于过度设计
- 对外api保持一致就OK
- 如果以后变化，整个全部重构即可

也有一些不坑人的好东东

案例7：消息系统的XMPP改进

第一版的问题

- 消息存放在ZODB
- 不断轮训
  - 性能影响
  - 实时性不够
- 更多实时协作：
  - 微博功能
  - 聊天

第一次尝试

- 基于gevent、rdb、socket.io
- 开发微博系统、实时消息推送
- 工程量庞大，中断：
  - 只是一个实习生做了demo
  - 产品化无人推动

选择xmpp

- 成熟的协议，广泛的支持、容易扩展
  - 服务器、客户端、开发包、示例代码、js
- 更多只是集成的工作
  - 实时的消息通知
  - 聊天，群组
  - 很容易增加类似的微博功能
- 自己的代码量非常少

具体技术栈

- xmpp服务器：prosody.im
- Xmpp Python客户端：Sleekxmpp
- Xmpp Js客户端：strophejs
- Web聊天窗口：conversejs
- 消息调试客户端：PSI

算是一次正确的选择

- 可以满足自己需求
- 未来可以调整
- 当然：学习理解xmpp协议是有门槛的

有耐心的演进史

案例8：易度3.0，4.0，5.0

易度PAAS演化

问题

- 易度办公应用，一度有10多个
- 其实低水平重复建设
- 代码非常多
- 维护成问题，比如想重构下
- 而且客户有各种新需求，不能满足

解决方法

- 构建PaaS开发平台
  - 吸取Zope的精华
  - 结合最新PaaS理念
- 构建应用市场
- 将内核应用，转到应用市场去

实际情况

- 历时2个版本，近一年的时间，逐步稳定
- 现有的10多个应用，逐步转移为扩展应用
  - 耗时3年
  - 最终代码量缩减一倍(更少的zope3代码！)
  - 系统变得很简单
- 系统可扩展性、可定制型加强
- 成为易度最独特的特色

有耐心的渐进迁移

- 暂停原有应用的大发展
- 发展新框架
- 等新框架成熟之后，数据迁移

易度OAUTH2和开放平台

API起始需求

- 客户向我们要api
- Webdav不完善
  - 协议复杂，基于xml
  - 功能不全
  - 各家实现不一致
  - 效率低下
- 手机版，也需要api

解决方法

- 使用oauth2，提供系统全套api

效果：很好！

- 系统架构更清晰
- 思考问题发生改变
- 无限可能
  - 内部需求：系统服务化，内部rpc调用
  - 桌面客户端的实现有新的方法
  - 将我们的云查看产品和文档管理统一了
  - 可以制作很多命令行工具
  - 摆脱对github的依赖
  - 和云查看的集成更简单

漫长的演进路

- web前端MVC改造：
  - Backbone
  - SPA
- 去zope3化
  - 更少的zope3
  - 可能也不着急

且做总结

- 这个时代的软件，没有停止的时候
- 我们别无选择：必须有耐心的演进
- 相信：我们总能找到一条前进通道
- 谨慎选择成熟的、标准的东西
- 做成熟的程序员，不要动不动想“干大事”
- 选择一个好语言：python